### CompSci 101
Introduction to Computer Science

<table>
<thead>
<tr>
<th>operation</th>
<th>mode 1</th>
<th>mode 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>pop</td>
<td>none</td>
<td>hybrid</td>
</tr>
<tr>
<td>pop</td>
<td>position</td>
<td>hybrid</td>
</tr>
<tr>
<td>sort</td>
<td>none</td>
<td>mutator</td>
</tr>
</tbody>
</table>

Feb 23, 2017

Prof. Rodger
Announcements

• Reading and RQ 11 due next time
• APT 4 is due on Tuesday

• Today:
  – More on Lists
  – Solving APTs with: while True
  – Coming – more ways to process data
  – Exam 1 back
Giving Back Exam 1… on Gradescope

• Solutions posted – regrades til March 3
  – Ask for regrade on gradescope
• Try working problem you missed first
  – Then look at solution

• Once you think you understand
  – Get blank sheet of paper – try again
• Understand all solutions
<table>
<thead>
<tr>
<th>Exam 1 scores</th>
</tr>
</thead>
<tbody>
<tr>
<td>86 86 86 86</td>
</tr>
<tr>
<td>85 85 85 85 85 85 85 85 85 85</td>
</tr>
<tr>
<td>84 84 84 84 84 84 84 84 84 84</td>
</tr>
<tr>
<td>83 83 83 83 83 83 83 83 83 83 83 83</td>
</tr>
<tr>
<td>82 82 82 82 82 82 82 82 82 82 82 82 82</td>
</tr>
<tr>
<td>81 81 81 81 81 81 81 81 81 81 81 81</td>
</tr>
<tr>
<td>80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80</td>
</tr>
<tr>
<td>79 79 79 79 79 79 79 79 79 79 79 79 79 79 79 79 79</td>
</tr>
<tr>
<td>78 78 78 78 78 78 78 78 78 78 78 78 77 77 77 77</td>
</tr>
<tr>
<td>76 76 76 76 76 76 76 76 76 76 76 76 76 76 76 76 76</td>
</tr>
<tr>
<td>75 75 75 75 75 75 75 75 75 75 75 75 75 75 75 75 75</td>
</tr>
<tr>
<td>74 74 74 74 74 74 74 74 74 74 74 74 74 74 74 74 74</td>
</tr>
<tr>
<td>73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73</td>
</tr>
<tr>
<td>72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72</td>
</tr>
<tr>
<td>71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71 71</td>
</tr>
<tr>
<td>70 70 70 70 70 70 70 70 70 70 70 70 70 70 70 70 70</td>
</tr>
<tr>
<td>69 69 69 69 69 69 69 69 69 69 69 69 69 69 69 69 69</td>
</tr>
<tr>
<td>68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68</td>
</tr>
<tr>
<td>67 67 67</td>
</tr>
<tr>
<td>65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65</td>
</tr>
<tr>
<td>64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64</td>
</tr>
<tr>
<td>63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63</td>
</tr>
<tr>
<td>compsci101 spring 2017</td>
</tr>
<tr>
<td>23</td>
</tr>
</tbody>
</table>
Problem:
Find the first duplicate adjacent words in a phrase, return the location of the first of the two words

- “Did I make make a mistake in in this?”
- Convert to list
- Answer is 2, (start counting at 0)
def positionDuplicate(phrase):
    words = phrase.split()
    if len(words) < 2:
        return -1
    pos = 0
    while True:
        pos = pos + 1
        return pos
APTs solved in a similar way with: while True

- Pikachu
- NameGroup
APT: Pikachu

Problem Statement

Pikachu is a well-known character in the Pokemon anime series. Pikachu can speak, but only 3 syllables: "pi", "ka", and "chu". Therefore Pikachu can only pronounce strings that can be created as a concatenation of one or more syllables he can pronounce. For example, he can pronounce the words "pikapi" and "pikachu".

You are given a String word. Your task is to check whether Pikachu can pronounce the string. If the string can be produced by concatenating copies of the strings "pi", "ka", and "chu", return "YES" (quotes for clarity). Otherwise, return "NO".

<table>
<thead>
<tr>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename: Pikachu.py</td>
</tr>
</tbody>
</table>
| def check(word):
  
  """
  return String based on parameter word, a String
  """

  # you write code here |
APT: Pikachu

• What is the iteration?
• What are the choices: pi ka chu

pichukarunkapi

Try:

Good:
APT NameGroup

1. "joe moe fred gus sam ted bo tom" "joe" "ted" 4

Returns "joe moe fred gus sam ted"

There is a name group that starts with joe, ends with ted and has 4 names between them.

2. "a b" "a" "b" 0

Returns: "a b"

There is a name group that starts with a, ends with b and has 0 names between them.

3. "a b c d b f g a b c d e f g a b c" "b" "g" 4

Returns: "b c d e f g"

Note that "b c d b f g" is not a name group since it contains b in the inner part. However, there is a valid name group, "b c d e f g", with no b nor g it.
Alice programming language
alice.org, Alice version 2.4
Nested Loop
Wac-A-Mole

World's details

<table>
<thead>
<tr>
<th>properties</th>
<th>methods</th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>moles</td>
<td></td>
<td>mole, mole2, mole3, mole4, mole5, mole6, mole7, mole8, mole9, mole10, mole11, mole12</td>
</tr>
</tbody>
</table>
Creating a list

• Given a list of numbers, create a second list of every number squared.

```python
nums = [8, 3, 5, 4, 1]
sqnums = []
for v in nums:
    sqnums.append(v*v)
print sqnums

[64, 9, 25, 16, 1]
```
More on List operations

• Previous page
  – `nameOfList “dot” function (parameter)`
    
    `sqnums.append(v*v)`

• See list operations on next page

• Mutator vs hybrid vs return
  – Mutator changes the list (no return value)
  – Hybrid changes list and returns value
  – Return – returns value, no change to list
### List operations from book

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Result</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>append</td>
<td>item</td>
<td>mutator</td>
<td>Adds a new item to the end of a list</td>
</tr>
<tr>
<td>insert</td>
<td>position, item</td>
<td>mutator</td>
<td>Inserts a new item at the position given</td>
</tr>
<tr>
<td>pop</td>
<td>none</td>
<td>hybrid</td>
<td>Removes and returns the last item</td>
</tr>
<tr>
<td>pop</td>
<td>position</td>
<td>hybrid</td>
<td>Removes and returns the item at position</td>
</tr>
<tr>
<td>sort</td>
<td>none</td>
<td>mutator</td>
<td>Modifies a list to be sorted</td>
</tr>
<tr>
<td>reverse</td>
<td>none</td>
<td>mutator</td>
<td>Modifies a list to be in reverse order</td>
</tr>
<tr>
<td>index</td>
<td>item</td>
<td>return idx</td>
<td>Returns the position of first occurrence of item</td>
</tr>
<tr>
<td>count</td>
<td>item</td>
<td>return ct</td>
<td>Returns the number of occurrences of item</td>
</tr>
<tr>
<td>remove</td>
<td>item</td>
<td>mutator</td>
<td>Removes the first occurrence of item</td>
</tr>
</tbody>
</table>

compsci101 spring 2017
Problem

• Remove all negative numbers from list
  \[4, -2, 5, 6, -3]\ \rightarrow\ [4, 5, 6]

• Two ways
  1) return a new list with all negative numbers removed
  2) Modify a list to remove negative numbers
```python
def removeNegatives(numberlist):
    answer = []
    for num in numberlist:
        if num >= 0:
            answer.append(num)
    return answer
```
somenums = [3, -1, 8, -5, -2, 6, 7]
nonegs = removeNegatives(somenums)
def removeNegatives2(numberlist):
    for x in range(len(numberlist)):
        value = numberlist[x]
        if value < 0:
            numberlist.pop(x)

somenums = [3, -1, 8, -5, -2, 6, 7]
removeNegatives2(somenums)
```python
def removeNegatives3(numberlist):
    pos = 0;
    while (True):
        if pos >= len(numberlist):
            break
        value = numberlist[pos]
        if value < 0:
            numberlist.pop(pos)
        pos = pos + 1
```

```python
somenums = [3, -1, 8, -5, -2, 6, 7]
removeNegatives3(somenums)
```
APT MorseCode

1. library = ["O ---", "S ..."]
message = "... --- ..."

Returns: "SOS"

The example from the problem statement.

2. library = ["O ---"]
message = "... --- ...

Returns: "?O?"

This time we don't have the S, so we replace the three dashes with question marks.

3. library = ["H -, "E .", "L --", "L ..", "O -."] message = "- . -- .. -." Returns: "HELLO"
Solving problems – APT MorseLikeCode

• Compare find vs index
  – find with string – returns -1 when not found
  – index with list – CRASHES if not there!
  – You can’t say: pos = alist.index(“…”)
  – Instead: if “…” in alist:
    pos = alist.index(“…”)

• How to get started?